Flutter Tutorial

Beginner to Expert

Table of Content

Introduction

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Windows, Mac, Linux, Google Fuchsia and the web.

Flutter is Google's mobile UI framework that can quickly build high-quality native user interfaces on iOS and Android. Flutter works with existing code. Flutter is being used by more and more developers and organizations around the world, and Flutter is completely free and open source . At present, some modules of the company are developed using Flutter
.
The major components of Flutter include:

Dart platform
Flutter engine
Foundation library
Design-specific widgets

Dart Platform
Flutter apps are written in the Dart language and make use of many of the language's more advanced features

You can refer Dart Language at Dart

Flutter Installation

Flutter is supporting HybridApp development on different Os.
To set up the flutter on each individual os by this [Flutter official Tutorial]

In this section we will learn how to install Flutter SDK in Windows system.

Step 1: Download Flutter SDK from [Official website] The Latest version is 1.12
Step 2: Unzip and archive file in specific folder, like c:\flutter\
Step 3: Update the system path to include flutter bin directory
Step 4: Open terminal and move to installed directory and run

```
flutter doctor
```

flutter doctor is tool to check all requirements for the flutter is installed or not and show the details

The result of the above command will give you

```
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel master, v1.14.6-pre.51, on Microsoft Windows
[Version 6.3.9600], locale en-IN)
[√] Android toolchain - develop for Android devices (Android SDK
version 28.0.3)
[√] Chrome - develop for the web
[√] Android Studio (version 3.3)
[!] Android Studio (version 3.4)
```

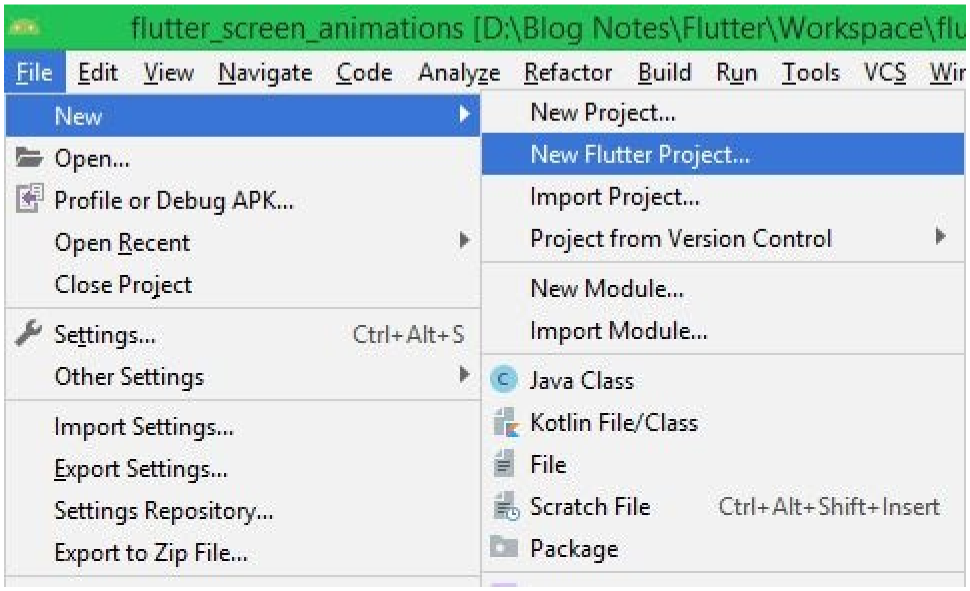
```
    X Flutter plugin not installed; this adds Flutter specific
functionality.
    X Dart plugin not installed; this adds Dart specific functionality.
[√] Connected device (2 available)

! Doctor found issues in 1 category.
```

Step 5: Install [Android Studio]()
Step 6: Check Latest Android SDK installed or not, if not install it
Step 7: Open Android studio and install Dart and Flutter Plugins for Android studio.

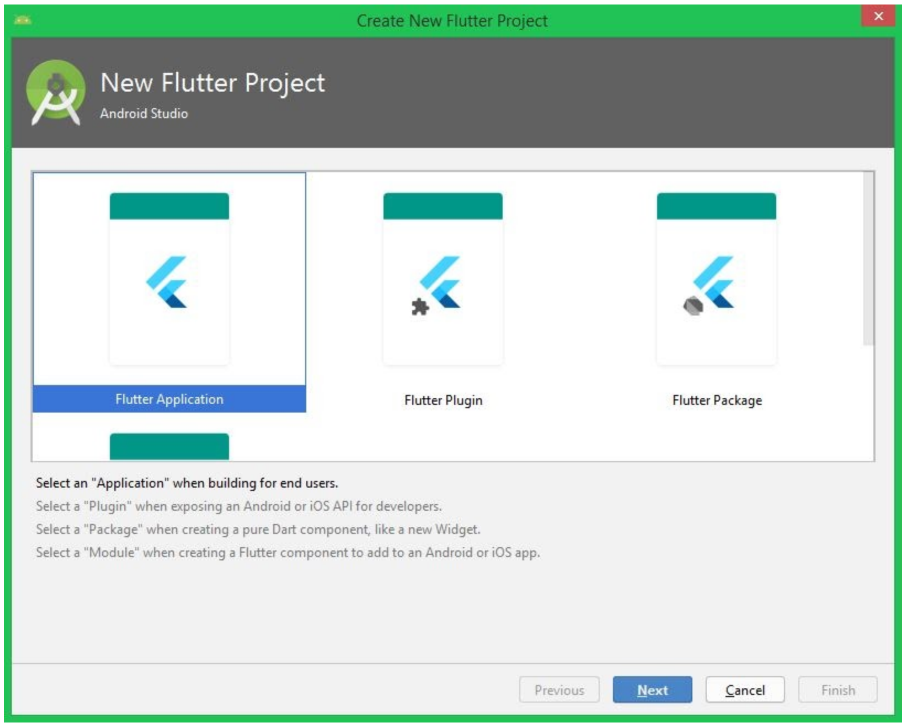
- Click File > Settings > Plugins.
- Select the Flutter plugin and click Install.
- Click Yes when prompted to install the Dart plugin.
- Restart Android studio

Flutter – Creating Simple Application in Android Studio

Now Open File -> Create new Flutter Project



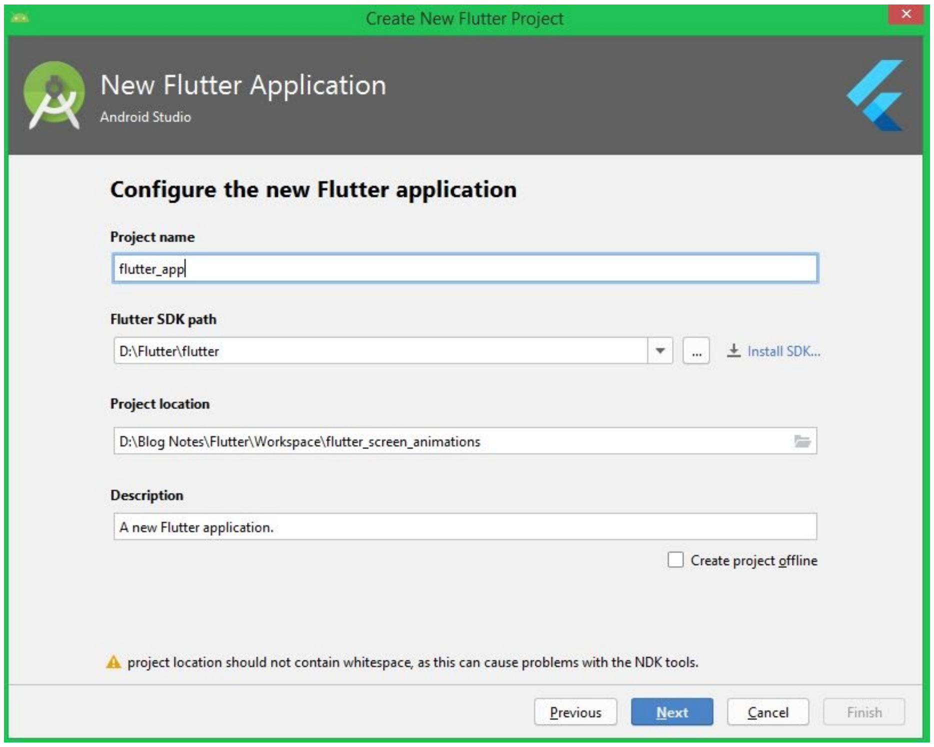
It will prompt below screen



Select an "Application" when building for end users.
Select a "Plugin" when exposing an Android or iOS API for developers.
Select a "Package" when creating a pure Dart component, like a new Widget.
Select a "Module" when creating a Flutter component to add to an Android or iOS app.

Select Flutter application and press Next

Now it will ask below details



Configure the new Flutter application

Project name: flutter_app

Flutter SDK path: D:\Flutter\flutter

Project location: D:\Blog Notes\Flutter\Workspace\flutter_screen_animations

Description: A new Flutter application.

[ ] Create project offline

⚠ project location should not contain whitespace, as this can cause problems with the NDK tools.

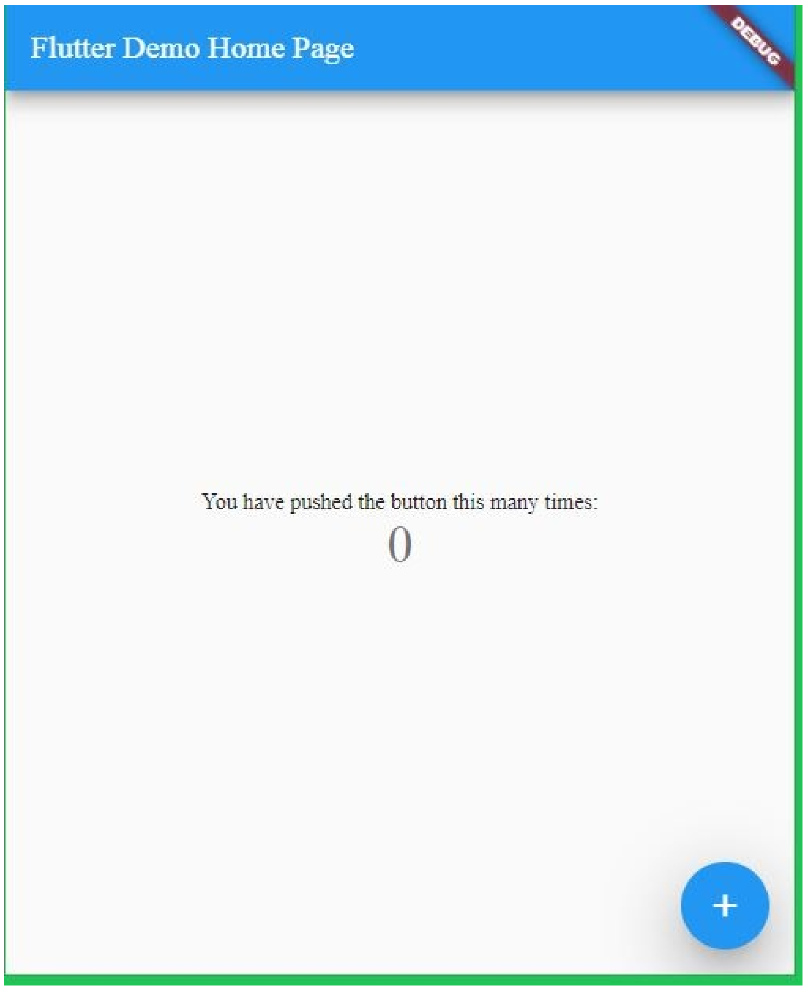
- Enter your Project Name
- Set Flutter SDk path (It is installation path)
- Set the Project Location
- Press Next Button

Enter Domain name and Press Finish

Now it will create a project with auto generated code

Now connect real device/Emulator and run the application

The output will be like this



You have pushed the button this many times:

0

Application Folder Structure

To understand flutter fully we need to understand the first Flutter folder structure.

- **android** - Auto generated source code to create android application
- **ios** - Auto generated source code to create ios application
- **web**- Auto generated source code to create web application
- **lib** - Main folder containing Dart code written using flutter framework
- **lib/main.dart** - Entry point of the Flutter application
- **test** - Folder containing Dart code to test the flutter application
- **test/widget_test.dart** - Sample code
- **.gitignore** - Git version control file
- **.metadata** - auto generated by the flutter tools
- **.packages** - auto generated to track the flutter packages
- **.iml** - project file used by Android studio
- **pubspec.yaml** - Used by Pub, Flutter package manager
- **pubspec.lock** - Auto generated by the Flutter package manager, Pub
- **README.md** - Project description file written in Markdown format

BLoC

Flutter, however, brings a new reactive style that is not entirely compatible with MVC. Its design idea is to separate data from views, and render views by data mapping

A variation of this classical pattern has emerged from the Flutter community – BLoC

What is BLoC?

BLoC stands for Business Logic Components, BLoC is a method of building applications using reactive programming. This is a completely asynchronous world composed of streams

- Wrap stateful components with StreamBuilder, streambuilder will listen for a stream
- This stream comes from BLoC
- The data in the stateful widget comes from the listening stream.
- User interaction gestures are detected and events are generated. For example, press the button.
- Call the function of bloc to handle this event
- After processing in bloc, the latest data will be added to the sink of the stream.
- StreamBuilder listens to new data, generates a new snapshot, and calls the build method again
- Widget is rebuilt

Example

Here we coding a simple counter application with BLoC

This Example show the Number counts in the first page, in the second page we are increase the counter number, this will reflect in the fistpage
For this we are going to create app with below steps

Create bloc model

CountBLoC

```
class CountBLoC{


 int _count = 0;
 var _countController = StreamController<int>.broadcast();

 Stream<int> get stream => _countController.stream;
 int get value => _count;

 addCount() {
   _countController.sink.add(++_count);
 }

 dispose() {
   _countController.close();
 }
}
```

Create Provider

```
class BlocProvider extends InheritedWidget{

 CountBLoC bLoC = CountBLoC();

 BlocProvider({Key key, Widget child}) : super(key: key, child: child);

 @override
 bool updateShouldNotify(InheritedWidget oldWidget) {
   // TODO: implement updateShouldNotify
   return true;
 }


 static CountBLoC of(BuildContext context) =>
      (context.inheritFromWidgetOfExactType(BlocProvider) as
BlocProvider).bLoC;
}
```

Firebase Integration

Firebase is a mobile and web application development platform developed by Firebase.

Firebase supports frameworks like Flutter on a best-effort basis.

Now we are going to learn how to setup firebase for Flutter Application

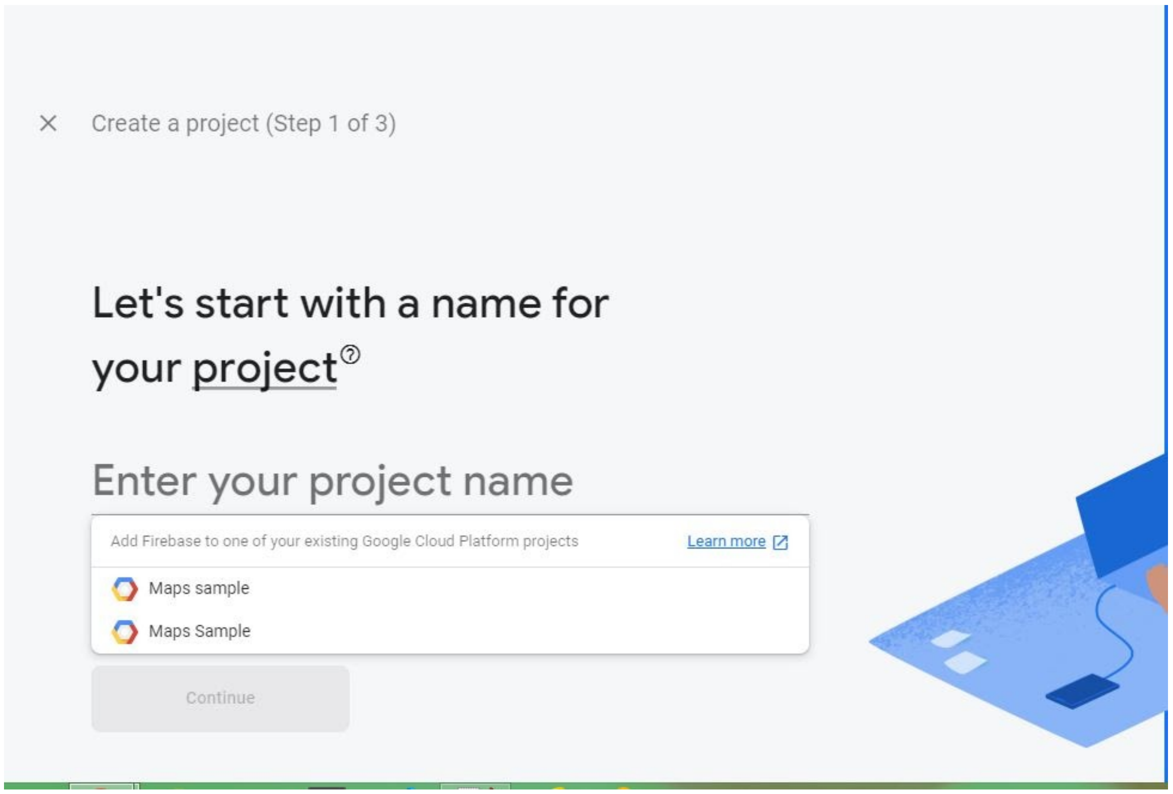
Firebase Setup

Step 1: Create a new Project in on firebase console (https://console.firebase.google.com/)

This will ask to login with gmail.

Login with your account.

Now create new project in Firebase console



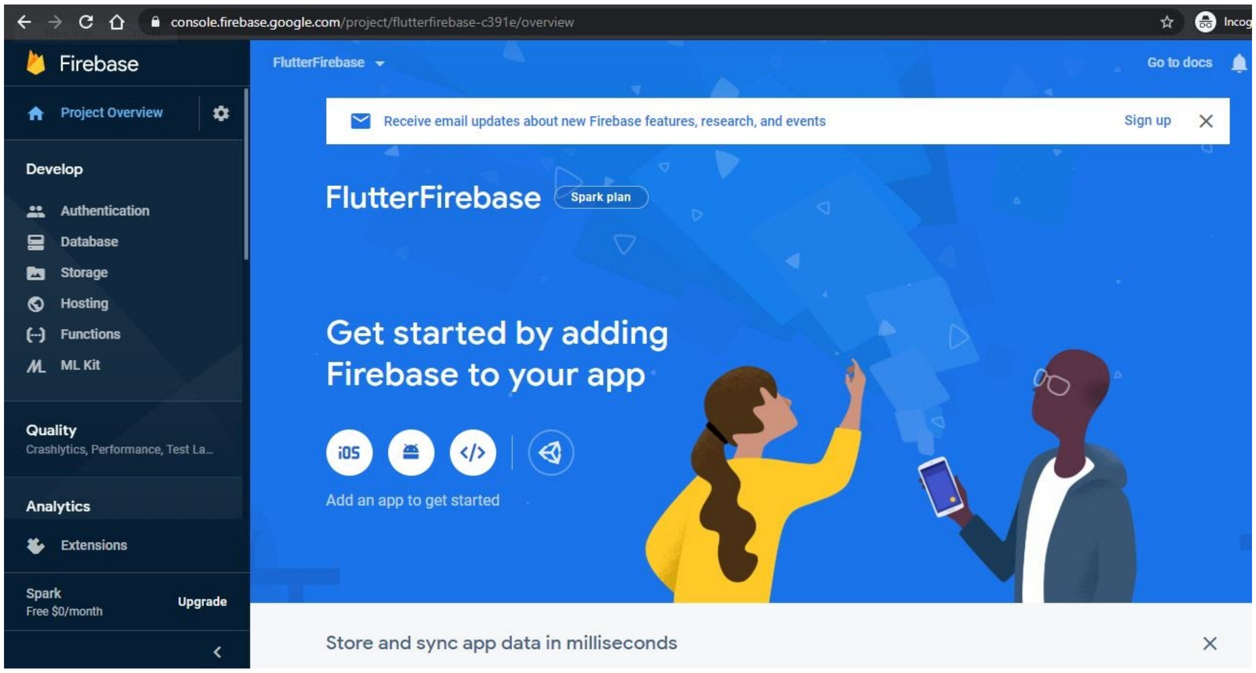
Create a project (Step 1 of 3)

Let's start with a name for your project

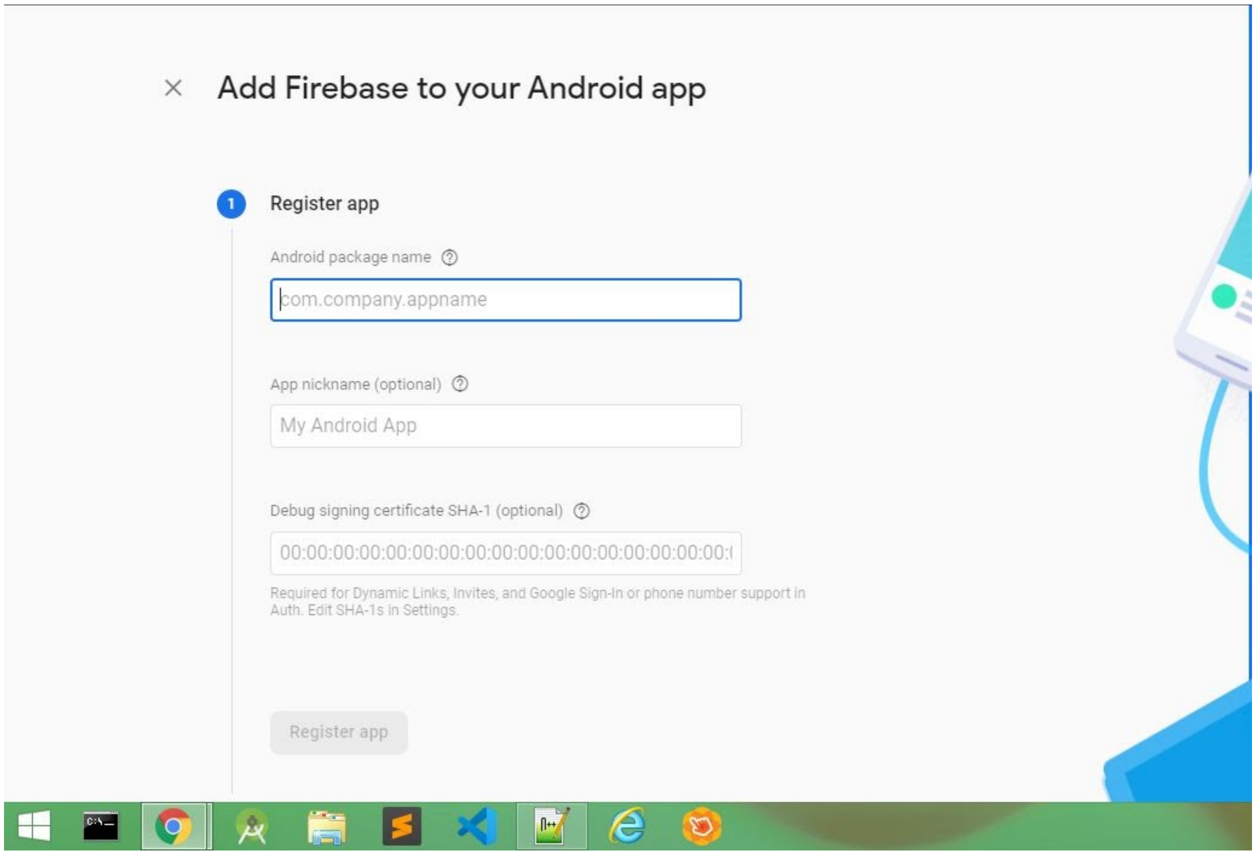
Enter your project name

Add Firebase to one of your existing Google Cloud Platform projects

After creating project it will navigates to Console dashboard


console.firebase.google.com/project/flutterfirebase-c391e/overview
Firebase
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test La...
Analytics
Extensions
Spark
Free $0/month
Upgrade
FlutterFirebase
Go to docs
Receive email updates about new Firebase features, research, and events
Sign up
FlutterFirebase
Spark plan
Get started by adding Firebase to your app
iOS
Add an app to get started
Store and sync app data in milliseconds

Now select Android, now it will display below screen


Add Firebase to your Android app

1 Register app

Android package name: com.company.appname

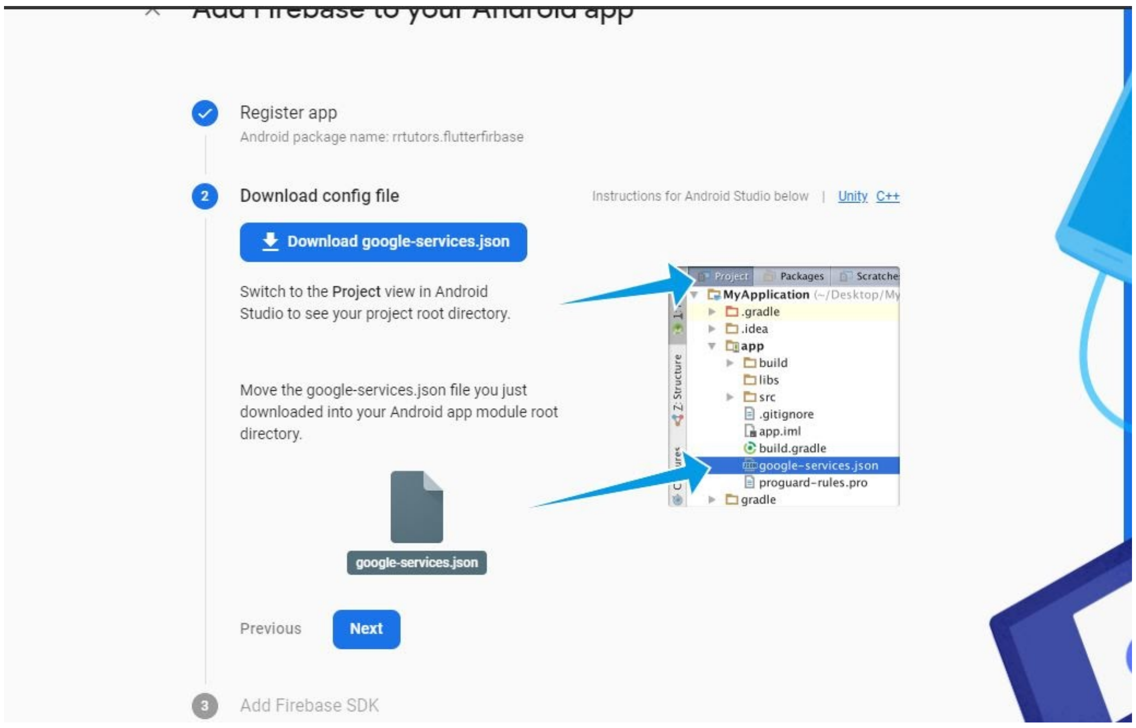
App nickname (optional): My Android App

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

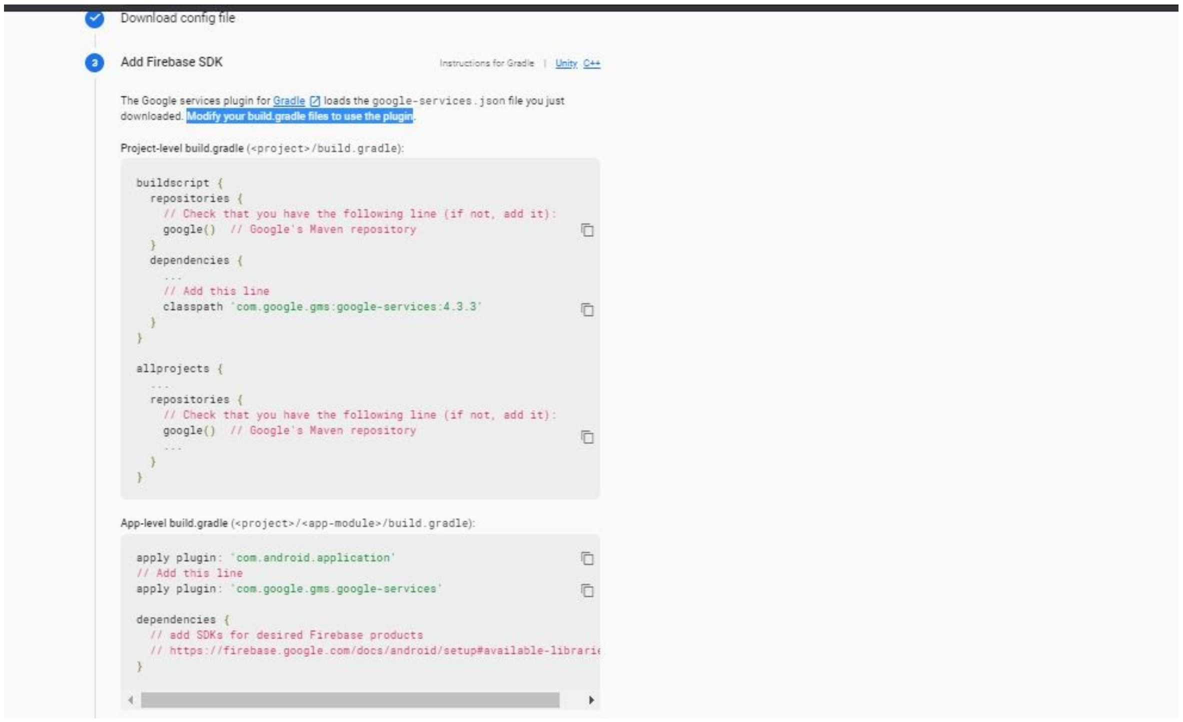
Add your application package name and register application



Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

Download google-service.json file and this under android->app folder



The Google services plugin for Gradle loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (<project>/build.gradle):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.3'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // add SDKs for desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-librarie
}
```

Modify your build.gradle files to use the plugin project level build.gradle file

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    // ...
    // Add the following line:
    classpath 'com.google.gms:google-services:4.3.3'  // Google Services plugin
  }
}
allprojects {
  // ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    // ...
  }
}
```

app level build.gradle file

```
  apply plugin: 'com.google.gms.google-services'
```

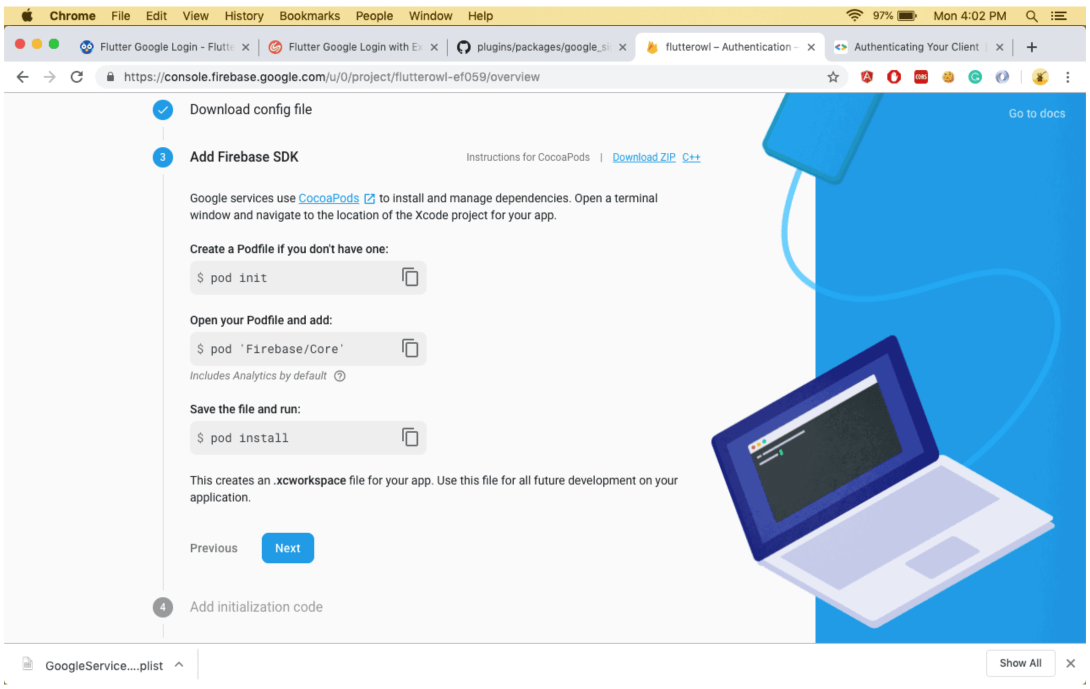
That's it for Android setup

iOS Setup

Copy & Paste your Downloaded GoogleService-Info.plist into projectname/ios/Runner folder
Open projectname/ios/PODFile (Execute pod install if file not found) & Add ;

```
   pod 'Firebase/Core;
```

and execute **pod install**


Chrome File Edit View History Bookmarks People Window Help
Mon 4:02 PM
https://console.firebase.google.com/u/0/project/flutterowl-ef059/overview
Go to docs
GoogleService....plist
Show All

Download config file

3 Add Firebase SDK

Instructions for CocoaPods | Download ZIP C++

Google services use CocoaPods to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don't have one:

```
$ pod init
```

Open your Podfile and add:

```
$ pod 'Firebase/Core'
```

Includes Analytics by default

Save the file and run:

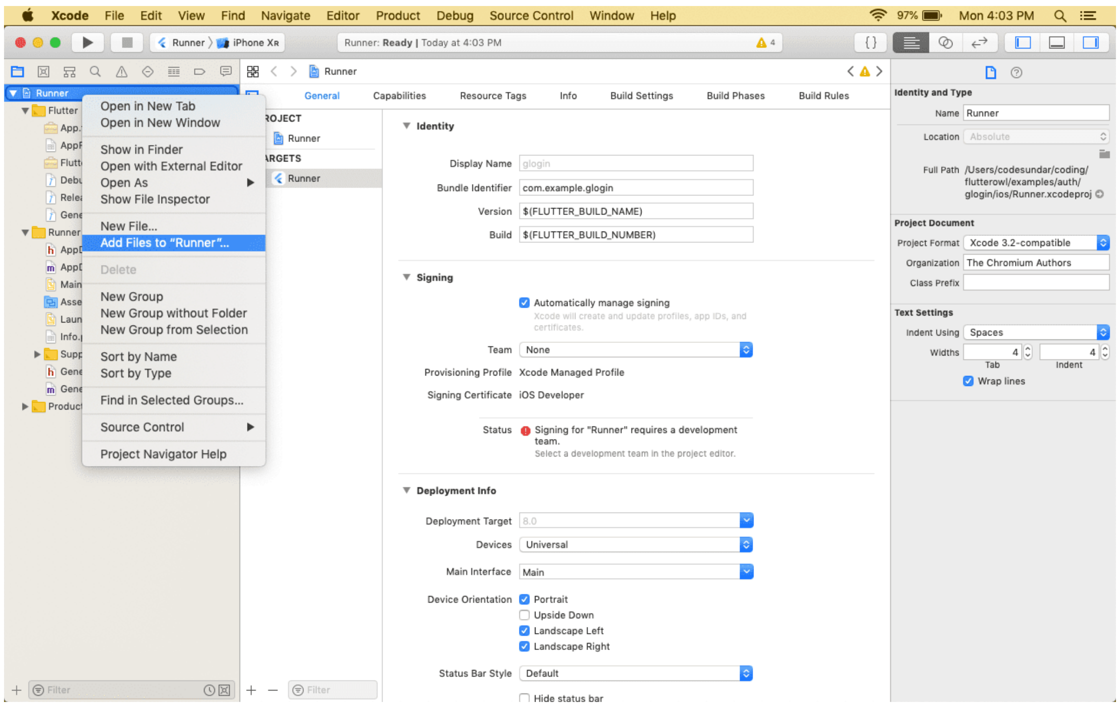
```
$ pod install
```

This creates an **.xcworkspace** file for your app. Use this file for all future development on your application.

Previous Next

4 Add initialization code

Open projectname/ios/Runner.xcworkspace & Choose Project & Add File to Runner & Choose GoogleService-Info.plist with choose target of Runner



Identity

Display Name	glogin
Bundle Identifier	com.example.glogin
Version	$(FLUTTER_BUILD_NAME)
Build	$(FLUTTER_BUILD_NUMBER)

Signing

✓ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team	None
Provisioning Profile	Xcode Managed Profile
Signing Certificate	iOS Developer
Status	Signing for "Runner" requires a development team. Select a development team in the project editor.

Deployment Info

Deployment Target	8.0
Devices	Universal
Main Interface	Main
Device Orientation	✓ Portrait ☐ Upside Down ✓ Landscape Left ✓ Landscape Right
Status Bar Style	Default

☐ Hide status bar

Identity and Type

Name	Runner
Location	Absolute
Full Path	/Users/codesundar/coding/flutterowl/examples/auth/glogin/ios/Runner.xcodeproj

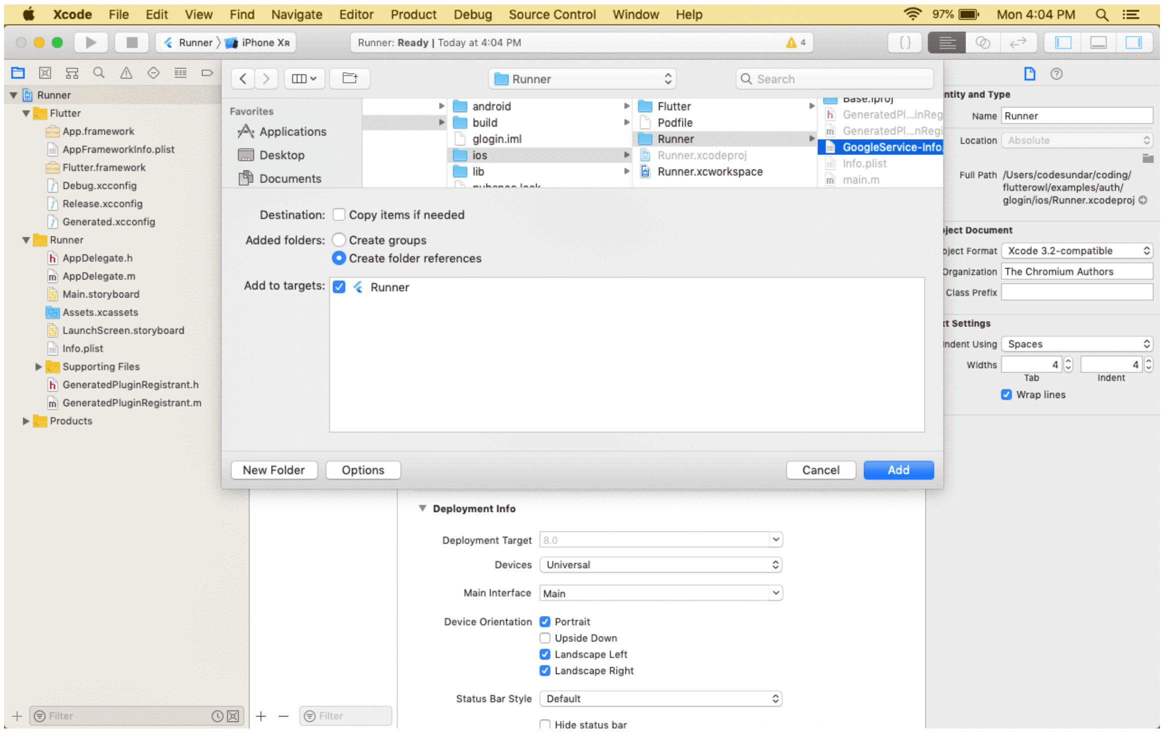
Project Document

Project Format	Xcode 3.2-compatible
Organization	The Chromium Authors
Class Prefix	

Text Settings

Indent Using	Spaces
Widths	4 Tab, 4 Indent

✓ Wrap lines



Destination: Copy items if needed

Added folders: Create groups [✓ Create folder references]

Add to targets: [✓ Runner]

Name: Runner

Full Path: /Users/codesundar/coding/flutterowl/examples/auth/glogin/ios/Runner.xcodeproj

Project Format: Xcode 3.2-compatible

Organization: The Chromium Authors

Indent Using: Spaces

Widths: 4 Tab, 4 Indent

[✓ Wrap lines]

Deployment Info

Deployment Target: 8.0

Devices: Universal

Main Interface: Main

Device Orientation: [✓ Portrait] Upside Down [✓ Landscape Left] [✓ Landscape Right]

Status Bar Style: Default

Hide status bar

That's it now our application is ready for use Firebase

Firebase authentication & Google sign in using Flutter

Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more

Now we are going to learn how to set up and implementing Google Signin using Firebase authentication

Chat Application with Firebase database Flutter